浮動小数点数の総和の計算法の比較 (計算の手間と能率化)

AUTHOR(S):

牛島, 和夫; 芦田, 正

CITATION:

牛島, 和夫 ...[et al]. 浮動小数点数の総和の計算法の比較 (計算の手間と能率化). 数理解析研究所講究録 1974, 215: 75-86

ISSUE DATE:

1974-07

URL:

http://hdl.handle.net/2433/105254

数理解析研究所講究録
215巻 1974年 75-86

浮動小数点数の総和の計算法の比較

九州大学工学部　牛島和夫
芦田　正

§1．はじめに

浮動小数点数の総和を求めることは基本的な数値計算の一つであるが、ふつうに使われている単純総和法（図1）では総和の中間結果が加数にくらべてかなり大きな値になつたとき、桁合せのために生じる打ち切り誤差の累積が無視できないものとなつてくる。そのため、単精度の浮動小数点数の総和の計算でもつと精度がほしいときには倍精度計算をするのがふつうであろう（図2）。

ところが、倍精度計算をしても、なおかつ誤差が問題となるような大量の計算が必要となる場合も考えられるし、

```
      S=0.0
      DO 1 I=1,N
      S=S+Y(I)
    1 CONTINUE
```

図1　単純総和法

```
      DOUBLE PRECISION S
      S=0.0
      DO 1 I=1,N
      S=S+Y(I)
    1 CONTINUE
```

図2　倍精度総和法

さらにハードウエアで倍精度浮動小数点演算の機能をもたない計算機もある。

ここでは、単精度計算だけを用いて計算時間をあまりかけずに総和をいかに精度よく求めるかという問題を考察する。

§2. 種々の浮動小数点数総和法

この問題に関して、いままでに種々の総和法が提案されているが、それらは大略2通りに分類できる。すなわち、第1は加える2数のオーダをそろえて桁合せのときに生じる打ち切り誤差を少なくしようとするもので、まずWolfe[1]がcascading accumulatorを用いた総和法を考案し、次いでそれをRoss[2]が改良してより柔軟性のあるアルゴリズムを与えた。しかし、彼らはいろいろな区間にある総和の中間結果をたくわえているcascading accumulatorを最後にどんな順序で加え合せたらよいかという点については詳しく論じていない。Malcolm[3]は、この問題を考察しWolfeのアルゴリズムをさらに改良するとともに詳しい誤差解析を行なつた。

次にLinz[4]はトーナメント方式による総和法を考案したが（図3）、その方法の最大の欠点は総和の計算をする前

に加えるデータがあらかじめ全部わかつていなければならないということであつた。しかし、この欠点も本質的なものではなく、Walker[5]はこの欠点を改良したアルゴリズムを与えた。

第2は、加える2数の桁合せのときに生じる打ち切り誤差を評価して補償項として加えてやろうというもので Kahan[6]（図4），Møller[7], Knuth[8], Dekker[9], Pichat[10],らにこの考え方が見られる。さらに、伊理、松谷[11] 平野[12]らの常微分方程式の数値解法における丸め誤差の取り扱いも同じ考え方にもとづいているといえる。

なお Malcolm のアルゴリズムは、データとして与えられた浮動小数点数をいくつかに分割して新しい浮動小数点数を作るという点に特長があるが、これをたとえば上位桁と下位桁に分割すれば Kahan の方法に近いものになる。したがつて、Malcolm の方法は厳密に分類すれば、第1と第2の考え方の中間に位置するものと考えられる。

以上は、算法の改良によつて総和の精度の向上をはかつたものであるが、誤差の評価という立場からは山下[13] が Wilkinson[14] の与えた誤差の上限を改良した結果を与えている。

§3．浮動小数点数総和法の比較実験

Gregory[15]はIBM 360/75 FORTRAN Hで Kahan の方法とLinz の方法を単純総和法と比較しているが、ここではこれらを含め次のような6通りのアルゴリズムを用いて総和の絶対誤差の絶対値の平均、標準偏差および総和に要する計算時間などを比較実験する。

P 1．　倍精度総和法　（図 2）

P 2．　単純総和法　（図 1）

P 3．　Linz の総和法　（図 3）

P 4．　Kahan の総和法（図 4）

P 5．　分類型 Linz の総和法

P 6．　Huffmanの総和法

以下、これらの方法をそれぞれ順にT，R，L，K，SL，Hと略すことにする。

```
  DO 2 J=1,K
  M=N/(2**J)
  DO 2 I=1,M
  Y(I) = Y(2*I)+Y(2*I-1)
2 CONTINUE
```

図 3　Linz の方法

```
  S=0.0
  S2=0.0
  DO 4 I=1,N
  S2=S2+Y(I)
  T=S+S2
  S2=(S-T)+S2
4 S=T
```

図 4　Kahan の方法

$(n = 2^k,\ k = \log_2 n)$

さて、上記のＰ５およびＰ６は比較実験のために付け加えた直観的に精度が向上すると思われるアルゴリズムで以下のとおりである。

Ｐ５．（分類型Linz の総和法）：　Linz のトーナメント方式で加えられる２数のオーダの差が大きければ、それだけ局所的な誤差は大きくなるし、正負の数がまざつているときは、そのほかに桁落ちという現象によつて精度が失なわれることもありうる。そこでLinz の方法を適用する前に、与えられたｎ個の浮動小数点数をあらかじめ小さい順に分類してトーナメントの左側に負数、右側に正数がくるようにすれば、桁落ちの起る可能性は大幅に減少し、したがつて直観的には総和の誤差も減少するように思われる。なお分類には、データの個数が 10^3 のオーダでもつとも分類時間が短いHoare　のQuicksort を用いた。

Ｐ６．(Huffman　の総和法）：　総和の局所的な打ち切り誤差を最小にするために、ｎ個の正の浮動小数点数の中から最小の数と２番目に小さい数を見つけて加え、次にその結果も含めたｎ－１個の正の浮動小数点数の中から、また最小の数と２番目に小さい数を見つけて加える。以下、同様の操作をくり返して最終的にはｎ個の正の浮動小数点数の総和を求める方法で、最適符号化にちなんで仮に Huffman

の方法と名付けた。ここでも、P5と同様にn個のデータを小さい順に分類することからはじめる。

さて、この実験で用いるデータの種類としては（0，1）の一様乱数(DATA 1),DATA 1のまん中に10^6のオーダの数をそう入したもの(DATA 2),DATA 1を底2で対数をとりその絶対値をとつたもの(DATA 3),DATA 1で偶数番目に正の符号を、奇数番目に負の符号をつけたもの(DATA 4)を考えた。データは1組1024個からなり、それぞれ単精度浮動小数点数として正しく表現されているものとする。すなわち、それぞれのデータは誤差を含んでいないものとする。

ここでは、データの種類ごとに20組のデータを各総和法毎に計算し、その絶対誤差の絶対値の平均、標準偏差および総和の計算だけに要した時間を比較した。ただし総和の真の値は同じ単精度データを用い図2の方法(T)で得られたものとする。

これらの計算はFACOM 230-60（2進、仮数部26ビット、4捨5入方式）およびFACOM 230-45S(16進、仮数部24ビット（6桁）、ガードデイジット付）でそれぞれコンパイルレベルOPT（最適化する）、NOOPT （最適化しない）で行なつた。この実験結果を表1－8に示す。なお、これらの実験でOPTとNOOPTの場合とも絶対誤差の絶対値の平

均、標準偏差は同じになつた。また時間の単位はmsec、時間比は単純総和法を基準にとつた。

§4. 結果の検討

表1 (FACOM 230-60, DATA 1)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT 時間	OPT 時間比	NOOPT 時間	NOOPT 時間比
T			17.3	1.28	24.2	1.03
R	0.375×10^{-4}	0.228×10^{-4}	13.5	1.00	23.6	1.00
L	0.248×10^{-4}	0.572×10^{-5}	16.1	1.19	61.5	2.61
SL	0.229×10^{-4}	0.384×10^{-5}	258.8	19.24	604.6	25.67
K	0.334×10^{-5}	0.201×10^{-5}	34.8	2.58	45.6	1.94
H	0.218×10^{-4}	0.441×10^{-5}	15758.8	1171.65	29352.9	1246.40

表2 (FACOM 230-60, DATA 2)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT 時間	OPT 時間比	NOOPT 時間	NOOPT 時間比
T			19.6	1.27	24.2	1.00
R	0.700×10^{-1}	0.374×10^{-1}	15.4	1.00	24.1	1.00
L	0.100×10^{-1}	0.638×10^{-2}	15.7	1.02	59.8	2.49
SL	0.113×10^{-1}	0.662×10^{-2}	280.7	18.22	573.9	23.86
K	0.395×10^{-2}	0.234×10^{-2}	34.7	2.25	44.0	1.83
H	0.395×10^{-2}	0.234×10^{-2}	15693.1	1019.03	26150.9	1087.36

表3 (FACOM 230-60, DATA 3)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT 時間	OPT 時間比	NOOPT 時間	NOOPT 時間比
T			19.1	1.29	23.9	1.00
R	0.976×10^{-4}	0.650×10^{-4}	14.8	1.00	23.9	1.00
L	0.642×10^{-4}	0.134×10^{-4}	16.9	1.15	61.5	2.58
SL	0.657×10^{-4}	0.160×10^{-4}	273.4	18.53	602.0	25.24
K	0.104×10^{-4}	0.491×10^{-5}	37.7	2.55	46.0	1.93
H	0.733×10^{-4}	0.105×10^{-4}	15949.1	1081.29	26239.1	1100.2

表4 (FACOM 230-60, DATA 4)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT 時間	OPT 時間比	NOOPT 時間	NOOPT 時間比
T			20.6	1.30	23.7	0.99
R	0.416×10^{-5}	0.433×10^{-6}	15.8	1.00	23.9	1.00
L	0.151×10^{-5}	0.227×10^{-6}	18.3	1.16	61.3	2.56
SL	0.222×10^{-4}	0.320×10^{-5}	320.1	20.26	585.7	24.51
K	0.711×10^{-6}	0.126×10^{-6}	45.0	2.85	48.2	2.02

表１－８を見ると、Ｋがいずれの場合も約１桁精度がよくＨ，Ｌがつづいている。DATA 2 に対しては、ＨはＫに匹適する。しかし計算時間を考慮すると他の方法にくらべて問題にならない。（Ｈの時間短縮については、大幅な改善は

表 5 (FACOM 230-45S, DATA 1)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT		NOOPT	
			時　間	時間比	時　間	時間比
T			22.6	1.43	32.6	1.24
R	0.656×10^{-1}	0.180×10^{-2}	15.8	1.00	25.8	1.00
L	0.595×10^{-3}	0.909×10^{-4}	17.9	1.13	34.3	1.33
SL	0.559×10^{-3}	0.617×10^{-4}	388.8	24.60	403.8	15.68
K	0.107×10^{-3}	0.631×10^{-4}	46.8	2.96	56.8	2.21
H	0.693×10^{-3}	0.138×10^{-3}	24025.59	1520.59	24049.5	933.96

表 6 (FACOM 230-45S, DATA 2)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT		NOOPT	
			時　間	時間比	時　間	時間比
T			22.7	1.43	32.8	1.27
R	0.160×10^{2}	0.382	15.9	1.00	25.9	1.00
L	0.313	0.682×10^{-1}	17.9	1.12	34.4	1.33
SL	0.353	0.387×10^{-1}	386.3	24.30	401.1	15.49
K	0.111×10^{-1}	0.152×10^{-1}	46.7	2.93	56.7	2.19
H	0.564×10^{-1}	0.197×10^{-2}	24025.3	1511.02	24025.3	927.62

表 7 (FACOM 230-45S, DATA 3)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT		NOOPT	
			時　間	時間比	時　間	時間比
T			22.5	1.42	32.4	1.25
R	0.103	0.195×10^{-2}	15.8	1.00	25.9	1.00
L	0.121×10^{-2}	0.141×10^{-3}	17.9	1.13	34.2	1.32
SL	0.121×10^{-2}	0.142×10^{-3}	385.5	24.40	400.3	15.49
K	0.255×10^{-3}	0.732×10^{-4}	46.6	2.97	56.9	2.20
H	0.117×10^{-2}	0.168×10^{-3}	21425.9	1356.07	21449.9	829.78

表 8 (FACOM 230-45S, DATA 4)

総和法	絶対誤差の絶対値の平均	絶対誤差の絶対値の標準偏差	OPT		NOOPT	
			時　間	時間比	時　間	時間比
T			22.3	1.44	32.3	1.28
R	0.340×10^{-3}	0.252×10^{-3}	15.5	1.00	25.4	1.00
L	0.565×10^{-5}	0.463×10^{-5}	17.9	1.15	34.3	1.35
SL	0.763×10^{-4}	0.777×10^{-4}	386.4	24.93	401.1	15.79
K	0.563×10^{-6}	0.360×10^{-6}	46.0	2.96	55.9	2.20

期待できない。）またSLはLよりも悪い場合が多く、とくに正負のデータが混合している場合はRよりも悪いことがある。これは、正負の数が同程度の個数あるときにトーナメントの最後の方で正と負の数を加えることによつて生ずる桁落ちの影響の方がトーナメントのはじめの方で生じた桁落ちの影響より大きいことを示している。つまり、小さい順にデータを分類することが逆効果になつて直観的な予想がくつがえされていることがわかる。さらに、SL，Hでは総和を求める前にあらかじめ加えるデータが全て求まつていなければならないという欠点がある。また表1－8を見ると、FACOM 230-60ではOPTの時間がNOOPTにくらべてかなり短かくなつている。FACOM 230-45Sでは、それほど減少していない。特に前者では、LとKの時間がOPTとNOOPTで逆転している。これはLの方がDOの制御変数を添字に使用しているために、最適化の効果が大きくあらわれたものと思われる。(FACOM 230-45SではNOOPTといえども添字の最適化をかなり行なつているからではないだろうか。）KがNOOPTにくらべてOPTの時間があまり減少していないのは、図4のプログラムから明らかなように、その方法自体が最適化の影響を受けにくいものであることを示している。さらに、T，R，L，Kはデータの性質によらないアルゴリズムであると

思われたが、表１－８を見ると、それぞれのデータの総和に要する時間が同じ総和法でも 2 msec 以上違つている場合もある。これは時間の測定単位が 1 msec であるという量子化の影響のほかに、ジョブの多重処理などの影響も考えられる。

§５．むすび

以上の検討より、精度と時間の両面から考えて Kahan の方法が一番よさそうである。しかし、この方法を盲目的に信用することは大変な危険性をはらんでいる。まず、この方法が有効であるためにはＡ＋Ｂを正規化してから単精度に丸める計算機でなければならないし、場合によつてはそのような処理系の性質をしらべるためにテストプログラム[16],[17]を用意する必要がある。

最後に、この計算は九大大型機センターの FACOM 230-60 および九大情報工学科の FACOM 230-45S を使用して行なつた。

参考文献

1. J.M. Wolfe: Reducing truncation errors by programming, Comm. ACM 7,6(June 1964), 355-356.

2. D.R. Ross: Reducing truncation errors using cascading accumulators, Comm. ACM 8, 1(Jan. 1965), 32-33.

3. M.A. Malcolm: On accurate floating-point summation, Comm. ACM 14, 11(Nov. 1971), 731-736.

4. P. Linz: Accurate floating-point summation, Comm. ACM 13, 6 (June 1970), 361-362.

5. R.J. Walker: Binary summation, Comm. ACM 14, 6(June 1971), 417.

6. W. Kahan: Further remarks on reducing truncation errors, Comm. ACM 8, 1(Jan. 1965), 40.

7. O. Møller: Quasi double precision in floating-point addition, BIT 5(1965), 37-50.

8. D.E. Knuth: Seminumerical algorithms, The art of computer programming, Vol.2, Addison Wesley(1968), 201-204.

9. T.J. Dekker: A floating-point technique for extending the available precision, Numer. Math. 18(1971), 224-242.

10. M. Pichat: Correction d'une somme en arithmetique a virgule flottante, Numer. Math. 18(1971), 400-406.

11. 伊理正夫，松谷泰行: Runge-Kutta-Gill 法について，情報処理 8, 2(1967), 103-107.

12. 平野泰彦： 常微分方程式の計算における丸め誤差の改善，情報処理14, 2(1973), 91-97.

13. S. Yamashita: On the error estimations in floating-point arithmetic, 学位論文(Oct. 1973).

14. J.H. Wilkinson: Rounding errors in algebraic processes, Prentice-Hall (1963), 1-33.

15. J. Gregory: A comparison of floating-point summation methods, Comm. ACM 15, 9(Sept. 1972), 838.

16. K. Ushijima: Preparations for formal study of a simple program including floating-point arithmetic, Rep. Compt. Centre, Univ. Tokyo, 4(1971-1972), 19-26.

17. M.A. Malcolm: Algorithms to reveal properties of floating-point arithmetic, Comm. ACM 15, 11(Nov. 1972), 949-951.